NORDIC MACHINE INTELLIGENCE

Nordic Machine Intelligence, vol. 1, pp. 1–15, 2022
https://doi.org/10.5617/nmi.9452

Hierarchical Object Detection applied to Fish Species

Espen Stausland Kalhagen [1], Ørjan Langøy Olsen, Morten Goodwin, and Aditya Gupta[2]

1. *E-mail any correspondence to: Espen.Kalhagen@gmail.com*
2. *Centre for Artificial Intelligence Research, University of Agder, Norway*

Abstract
Gathering information of aquatic life is often done manually using video feeds. This is a time consuming process and it would be beneficial to capture this information in a less laborious manner. Video based object detection has the ability to achieve this. Recent research has shown promising results with the use of YOLO for object detection of fish. Detecting fish underwater is difficult because of low visibility and therefore fish species can be hard to discriminate. To alleviate this, this study proposes the fish detector YOLO Fish, which uses hierarchical techniques in both the classification step and in the dataset. Hierarchical techniques handles situations where features are difficult to discern better and makes it possible to extract more information from the data. With an mAP of 91.8%, YOLO Fish is a state-of-the-art object detector on Nordic fish species. Additionally, the algorithm has an inference time of 26.4 ms, fast enough to run on real-time video on the high-end GPU Tesla V100.

Keywords: Object Detection; YOLO; Fish Species; Hierarchical Classification

Introduction

Object detectors can face difficulties when dealing with classes from certain biological domains because there can be large similarities between species from a purely visual standpoint. Further, with the lack of clear and visible features under water, as compared to on land, the object detector faces difficult operating conditions. In the biological domain the taxonomic hierarchy specifies species and their relation to each other. This system of relations can aid in the classification of species as the hierarchical nature can be exploited to face the difficulties of hard to distinguish objects. The detection of fish in video feeds is a good example of a domain where species can be similar-looking and conditions difficult. Currently information gathering of aquatic life is done manually with expensive techniques and limited information, often with a basis in video feeds [1][2]. Collecting information about aquatic life is highly important for sustainable and profitable marine life management [3][1]. The use of machine learning techniques such as object detection and classification can aid with this.

Hierarchical Classification

Huang, Boom, and Fisher [4] exploited the hierarchical nature of fish features to increase classification accuracy on the fish4knowledge dataset [5]. Using a Balance-Guaranteed Optimized Tree for constructing a hierarchy and a SVM at each level of the hierarchy they achieve a classification average precision of 91.7% on a dataset containing 3179 fish from 10 classes. Redmon and Farhadi [6] utilises global hierarchical classification with Darknet-19 on the ImageNet dataset by transforming the WordNet into a tree structure called WordTree. This classifier achieves 71.9% accuracy on a subset of ImageNet containing 1000 leaf classes, or 1 369 with the classes needed in the hierarchy.

Object Detection of Fish Using CNN

Xiu Li et al. [7] did object detection of fish on a dataset based on fish4knowledge and used Fast R-CNN to achieve a mean Average Precision (mAP) of 81.4% at an inference time of 311 ms per image, and Fast R-CNN with singular value decomposition to achieve an mAP of 78.9% at 273 ms per image. This is further improved in 2017 by Li, Tang, and Gao [8] which achieved a 89.5% mAP at 89 ms per image. Raza and Hong [9] did object detection of fish using a modified version of YOLOv3 on their dataset with 4 classes of aquatic life with one of them being for fish. By modifying the loss function and adding an extra detection scale to account for finer-grained features they achieve an mAP of 91.3%. Object detection of salmon in Norwegian fish farms has been done by Reithaug [10] where using SSD Inception V2 an mAP

of 84.64% is achieved at 266 ms seconds per image.

Classification of Nordic fish has been done by Olsvik et al. [11] where they attained an accuracy of 87.74% with a CNN-SENet and 90.20% with a ResNet-50 on their dataset containing 867 fish across 4 classes. The classifier from Olsvik et al. [11] is expanded for object detection using YOLOv3 by Knausgård et al. [12] where YOLO is used for localization only, and classification is done with CNN-SENet. The localizer achieves an mAP of 87% (without classification error) on their data. The classifier achieves a classification accuracy of 83.68% on the dataset of Knausgård et al. [12].

You Only Look Once (YOLO)

YOLO [13] is a state-of-the-art algorithm for object detection. The algorithm has seen some incremental improvements since its first release with accompanying published papers. The versions are called YOLOv1 [13], YOLOv2 (YOLO9000) [6] and YOLOv3 [14]. There has been some significant changes from one version to the other and are briefly summarized as follows. YOLOv2 is predicting how to move and scale some prior anchor boxes relatively, instead of directly predicting bounding boxes [6]. YOLO9000 added the WordTree concept to predict classes that are hierarchically organized and was designed to be able to classify over 9000 classes [6]. YOLOv3 added multi-class and multi-scale prediction [14]. YOLOv3 does not have the concept of a WordTree.

Drawbacks with previous work

When you label a localization and classification dataset there will often be fish in the frame that are difficult to discern the species of. Previous approaches often utilize a class representing unknown fish where fish species cannot easily be discerned to solve this problem. A downside with this is that the algorithm specifically learns that some fish are of the class 'unknown'. This gives them the ability to include fish they don't know the species of. However, it is reasonable to believe, that the detector learns that less visible fish are of the class 'unknown' and not that this class can be applied when it is able to localize, but not classify.

Objective

In this project, the goal is to create a new object detector called YOLO Fish that utilizes the taxonomic hierarchy to detect fish in their natural habitat. To achieve this a new manually annotated dataset needs to be created.

The following list summarizes the objectives.

- Create an end-to-end model for real-time localization and classification of Nordic fish species in video.
- Achieve state of the art performance on real-time localization and classification of Nordic fish in video.
- Use the taxonomic hierarchy for data annotation to make data generation significantly easier for difficult domains.
- Create a method that enables training and detection in conditions where it is not always possible to discern species.

Method

Hierarchical Classification

Most convolutional neural networks use a flat structure for classification, but a hierarchical structure can also be useful when modeling domains that are inherently hierarchical.

This structure can be designed such that each node has a "is a" relationship to its parent. For example, a fish is an animal, so thus 'fish' can have 'animal' as a parent.

YOLO's WordTree uses hierarchical probabilities to apply the most specific class (furthest down in the tree) that still is above some hierarchical threshold. This is done by calculating the conditional probabilities $P(\text{Node}|\text{Parent})$ for each node in the WordTree starting with the root. It calculates this recursively down the tree, following the node with the highest probability. At the point where the calculated probability is lower than the hierarchical threshold the recursion is stopped, and the parent class is returned. As a result the most specific node that can confidently be selected is returned and the probability of this node is based on itself and its parents. [6]

Most implementations of neural networks applies softmax across all classes to get a probability distribution, resulting in a single-label predictor. To be able to do hierarchical classification efficiently YOLO's WordTree applies multiple instances of softmax across siblings. This makes each level stand alone and makes it possible for the detector to discriminate classes accurately. [6]

Data Generation Strategy

Object detection in images and video require data in the form of images with accompanying bounding boxes with assigned classes. Compared to image classification tasks, this requires a lot of manual labour as this kind of data is not easy to come by. Additionally, to be able to test hierarchical classification, the dataset needs to be created with hierarchical classes because the hierarchy affects what labels are applied. This is because the algorithm needs to learn what features can distinguish certain species, and what features are common among them. Therefore, the class needs to reflect the visible features of the fish. By using hierarchical classification with a dataset specifically created for this, the network can mimic the mental model humans have of fish and fish species. This makes it possible for it to apply classes based on the features that are visible, resulting in classifications that are as specific as possible.

Redmon and Farhadi [6] use hierarchical classification to combine two incompatible datasets. Just applying hierarchical classification without a dataset that really leverages this works to combine the datasets, but is not sufficient when the goal is to use this to account for

the lack of visible features, and gaining more valuable information from the data.

Because of the previously mentioned reasons, a custom dataset and data generation strategy is required. The images are collected from an underwater video stream and imported into an image labeling tool. To ensure the dataset is consistent, a set of strategies are established on how to annotate the data that is rooted in the theory behind YOLO. These are specifically based on loss handling.

The following approach is defined for annotation.

1. Annotate all fish that are visible to humans, without the context of neighboring frames.

2. Apply the most specific species label that can confidently be discerned.

The loss function penalizes detection where it falsely detects fish. Thus a goal is to make sure that all the fish that can be detected are annotated with a bounding box.

On the other hand, with the use of a tree structure for classes, the annotation of classes has seen the use of the opposite tactic. Only the most specific taxon that can very confidently be applied to the fish is utilized. This strategy is rooted in the design of YOLO where loss is only applied to the ground truth class and the above classes in the WordTree [6]. This does not penalize the model for predicting a more specific class than what is annotated. An example of this can be seen in Figure 1, where *Pollachius pollachius* has been predicted, but the ground truth specifies Gadidae, so only the classes Gadidae and "Fish" have their loss backpropagated. This increases the confidence in the network's ability to predict the correct classes since the confidence of the class in the dataset is very high. It also makes it significantly easier to create the dataset since not all fish in an image needs to have a species specified.

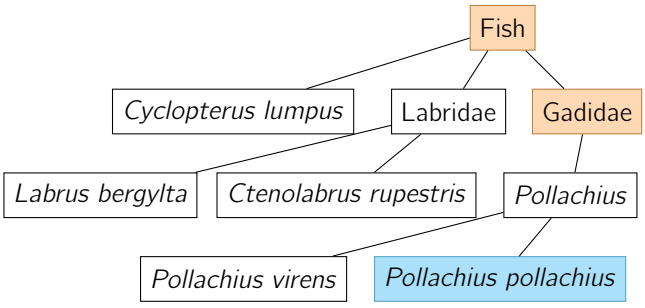


Figure 1: The classes used in YOLO Fish. If the network predicts *Pollachius pollachius* (blue), but the ground truth is Gadidae, only the orange classes Gadidae and "Fish", have their loss backpropagated.

Dataset

A new dataset is required for this project as no publicly available datasets fits the needs of the project. No large object detection dataset with multiple fish in each image was found in literature. Additionally, to be able to show the effectiveness of hierarchical species classification, a dataset with a hierarchical labeling strategy is required. As part of this work, such a dataset was created and annotated. This dataset is annotated by the authors and thus may lack the rigour of a biologist. The dataset is made publicly available[1] with the trained models so that they can be used for transfer learning on downstream tasks.

The dataset contains a collection of underwater images of fish, and annotation files that specify the bounding box of each fish in the image, and a class. These are specified in the YOLO format. The images have a resolution of 1920x1080 and are taken from a camera that is located in Lindesnes, Norway. The pictures were captured in a period between February and March 2020. The dataset consists of 1879 images and corresponding annotation files. There are 7721 labeled fish in total and are distributed over 7 hierarchical classes as shown in table 1. The test set contains 10% of the images which equates to 188 images and the training set contains 1691 images. Each image has a varying number of fish in each image.

Table 1: Distribution of classes in the dataset.

Class	Count
Fish	5810
Gadidae	791
Pollachius virens	537
Ctenolabrus rupestris	212
Pollachius pollachius	172
Cyclopterus lumpus	147
Labrus bergylta	52
Labridae	0
Pollachius	0
Total	7721

The images contain fish in motion and from many angles. This makes the fish in the images have very varied shapes, sizes, and visible features, as seen in figure 2. The fish can mostly be seen swimming above, towards, away, or parallel to the camera because of its location on the bottom of the seabed. The pictures were captured in day, dusk, night, and dawn, and in both clear and turbid waters and therefore provides a wide variety of conditions. There is also a significant portion of the frame where seaweed can be seen and many of the pictures contain fish in the seaweed. This is a quite difficult dataset in contrast to for example fish4knowledge. This dataset more represents the real world's noisy and imperfect conditions that a real application would be exposed to. During the period

[1]The new dataset can be found on https://dx.doi.org/10.17632/b4kcw9r32n.1

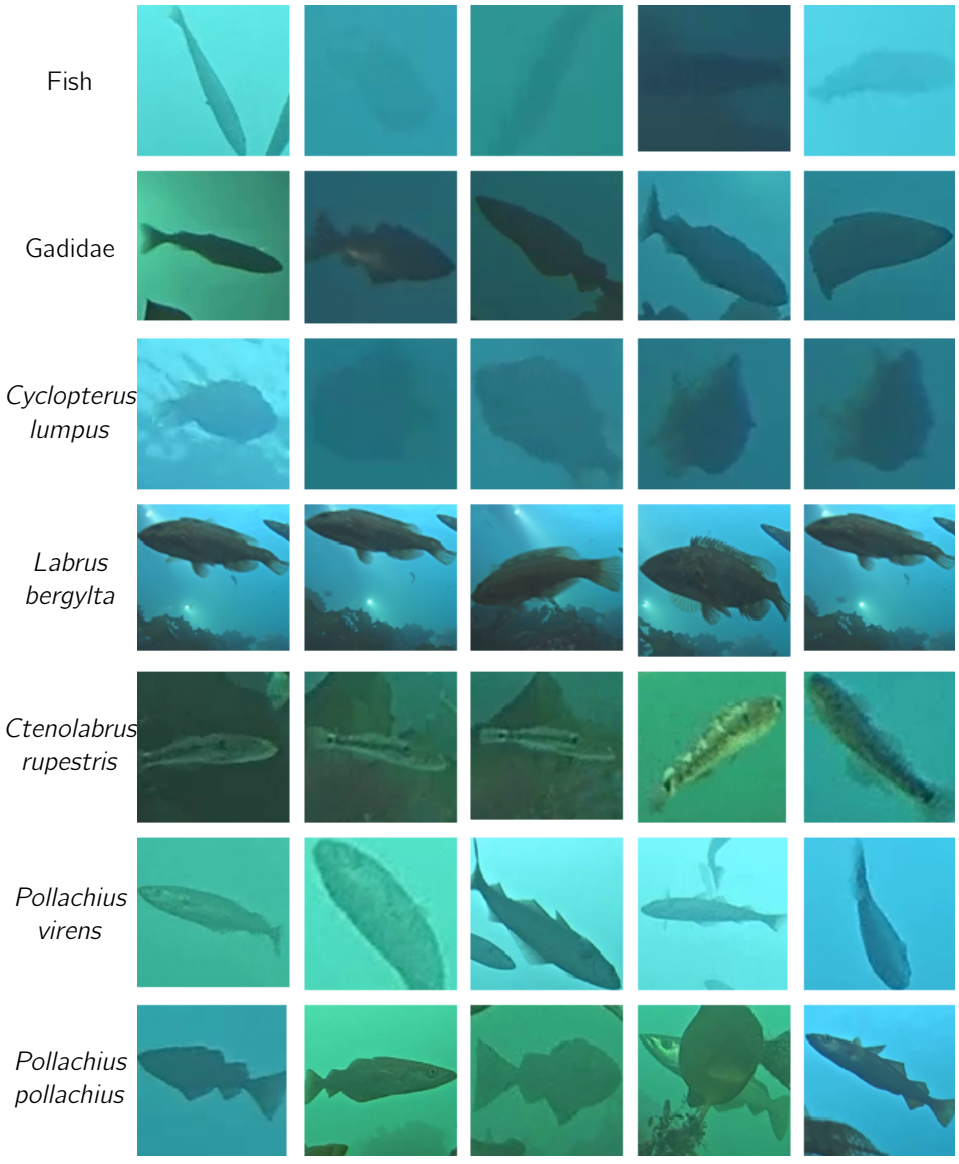


Figure 2: Examples of fish in the dataset from the different classes.

when the images are captured, there is an abundance of *Pollachius pollachius* and *Pollachius virens*, which is of the *Pollachius* genus and Gadidae family. This means that over-sampling techniques are often utilized for other species. Thus these other species do not have labels in as many varied conditions, but hierarchical classification is still capable of picking up on useful information.

Evaluation

Selecting appropriate evaluation measures is important to be able to accurately assess the performance. The standard measure for evaluating the performance of object detectors is mAP [15], because of its use in prominent competitions like Pascal VOC [16].

For experimental evaluation, multiple different metrics are used. Optimal Localization Recall Precision (oLRP) is used as a metric for assessing the performance of the network together with mAP. This is because it does not encounter the same problems as mAP, and has strengths that makes it accompany mAP well. Oksuz et al. [15] notes several problems with the standard performance measure, mAP. The most important being:

- Unable to distinguish different precision-recall (PR) curves.
- Lack of directly measuring bounding box localization accuracy.
- Uses interpolation between neighboring recall values, which can can misrepresent the actual results. [15]

Therefore Oksuz et al. [15] proposed a new performance metric called Localization Recall Precision (LRP). LRP is made up of three parts, a bounding box error part, a false positive (or precision) part, and a false negative (or recall) part. It is therefore able to represent both precision and recall errors, but also bounding box errors. It does this without calculating the area under the PR curve, hereby avoiding the need for interpolation. As opposed to mAP, a lower LRP value indicates better performance. LRP is defined in equation 1.

$$\mathrm{LRP}(X, Y_s) = \frac{\sum_{i=1}^{N_{\mathrm{TP}}} \frac{1-\mathrm{IoU}(x_i, y_{x_i})}{1-\tau} + N_{\mathrm{FP}} + N_{\mathrm{FN}}}{N_{\mathrm{TP}} + N_{\mathrm{FP}} + N_{\mathrm{FN}}} \tag{1}$$

where τ is the IoU threshold for predictions being considered correct, and where N_{TP}, N_{FP}, and N_{FN} is the number of true positives, false positives and false negatives respectively.

In addition Oksuz et al. [15] introduced oLRP which is the minimum achievable LRP error. This lowest error is achieved at some probability threshold, which is denoted s. oLRP represents the best achievable bounding box-localization and recall for the detector. oLRP is defined in equation 2, where predictions with an IoU greater than 0.5 are considered correct. Reported precision and recall are calculated at the probability threshold s.

$$\mathrm{oLRP} = \min_s \mathrm{LRP}(X, Y_s) \tag{2}$$

Architecture Improvements

YOLO cannot be used out of the box because it is not tailored for the problem, and therefore has received various changes to become YOLO Fish. The baseline architecture is YOLO9000 because it supports hierarchical classification. Starting with YOLO9000, the following significant improvements are performed until ending up with the fish detector YOLO Fish.

- Modified Non-Maximum Suppression (NMS) to account for hierarchical predictions.
- Replaced YOLOv3 detection layers with YOLO9000 detection layers.
- Incorporated Soft-NMS.

The NMS algorithm only removes overlapping bounding boxes of the same class. When using hierarchical classification an object can be of several classes across the levels in the hierarchy at the same time. This causes a problem when used together with class dependant NMS as it does not remove objects where the classes are at different levels in the hierarchy. To solve this problem NMS has been modified to remove all overlapping boxes regardless of class. This can be seen in figure 3.

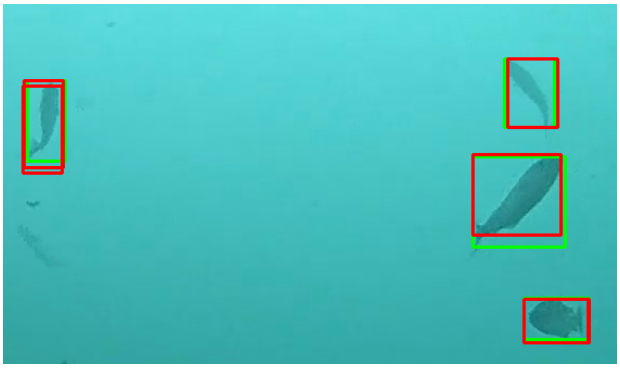
NMS now removes all overlapping bounding boxes, hence too many bounding boxes may sometimes be removed. Therefore Soft-NMS [17] is implemented to test the improved performance. Soft-NMS gradually decays detection scores as the Intersection over Union (IoU) increases. Thus instead of setting the probability to 0, the probability is now calculated using equation 3. This way no bounding box is directly given a probability of 0. These can potentially be removed later with a probability threshold, as done when calculating oLRP, or before drawing bounding boxes as is common practice.

$$\mathrm{SoftNMS} = (1 - \mathrm{IoU}) * P_c \tag{3}$$

where P_c is the probability of class c as outputted by YOLO.

Training

The task is fish species detection, hence the network needs to be trained on new data. All the networks presented in this article are trained in the same way with the same parameters and settings, except for the varying use of the WordTree and network resolution. Using transfer learning does not fit with the experiment setup as it would make the training different between networks, and



(a) Multiple bounding boxes per fish.

(b) A single bounding box per fish.

Figure 3: Predictions from YOLO before (a) and after (b) modification of NMS. Green bounding boxes indicate correct predictions, red indicate wrong.

thus their performance less comparable. The fish in the dataset very often share the same aspect ratios and sizes, since the camera is in the same position and fish tends to swim in the same ways. Therefore computing good anchor boxes significantly helps the localization. Nine anchor boxes that fit the training set are computed with K-means. Nine anchor boxes are selected because more anchor boxes give diminishing returns in IoU overlap and an increased inference time. Fewer bounding boxes however reduces the IoU overlap significantly, as seen in Table 2. In the case of YOLOv3 each detection layer is responsible for three of the anchor boxes.

Table 2: Increasing the number of anchor boxes beyond nine yields diminishing returns.

Anchor boxes	6	9	12
Avarage IoU	0.63	0.69	0.71

In addition to new anchor boxes some other training parameters are important to note. Letter boxing is used to keep the image aspect ratio. A network input resolution of 608x608 is used. While there are only seven classes of fish in the dataset as noted in the Dataset section, intermediate classes are added to allow for the tree structure. The tree structure with all the classes is shown in figure 1. To keep things as similar as possible everything is trained with the nine classes, even if WordTree is not used. In addition these image augmentations are randomly applied during training:

- Image resizing
- Hue change
- Saturation change
- Exposure change
- Image cropping (jitter)

Experiments and Results

Experiments on the Hierarchical Classifier

When creating datasets that can be used for object detection and species classification of fish it is difficult to correctly assign classes to all the fish in an image. This is because an image is likely to contain clearly visible fish in the foreground and less visible fish in the background. Some datasets uses images that are cropped to fit a specific fish of interest, but this cannot be used to effectively train an object detection algorithm. To solve this, fish can be labeled using the taxonomic hierarchy, making it possible to always apply labels confidently. This experiment tests whether it is possible to train YOLO, using YOLO's WordTree in combination with data labeled with taxa from the taxonomic hierarchy. And whether YOLO can detect fish and correctly assign classes when it is trained on a dataset that contains fish labeled using this hierarchy.

Testing The Effectiveness

To verify the effectiveness, the YOLO9000 algorithm is trained on the dataset that contains hierarchically labeled fish, and the resulting model evaluated. Table 3 shows the model's performance on validation data, and presents the number of fish labeled more and less specific than the ground truth. This makes it possible to see that the model is able to correctly assign the most specific class it can, over a probability threshold.

Table 3: YOLO9000 with hierarchically predicted classes.

Species	Fraction Correct	More specific	Less specific
Fish	501/653	500	0
Gadidae	76/81	76	0
Cyclopterus lumpus	12/14	0	0
Labridae	0/0	0	0
Pollachius	0/0	0	0
Labrus bergylta	6/6	0	0
Ctenolabrus rupestris	23/27	0	0
Pollachius virens	65/69	0	65
Pollachius pollachius	14/14	0	14

Table 3 shows a detailed view of classifications per class. Column two shows the proportion of detected fish that is classified correctly for that label. Column three shows the number of fish classified correctly and that has been applied a more specific class in the hierarchy than the ground truth data. Column four shows the number of fish classified with a less specific class, but a class still in its branch of the tree.

Table 3 shows that all Gadidae that were detected is assigned a more specific class than the ground truth data. Consulting Figure 1 shows that Gadidae has multiple sub taxa with which it shares many features. This means the network can learn feature information from other training examples and transfer this into being correct for the Gadidae class, and then apply that class. The same goes for *Pollachius virens* and *Pollachius pollachius*, where the network has applied less specific classes compared to ground truth data, but still a parent class. Comparing those classes in Figure 1, it becomes apparent that there are few distinguishing features. The network has then not been confident enough to distinguish between the classes and then applied a less specific class, for instance *Pollachius*.

Testing The Performance

By using YOLO's WordTree it is possible to generalize data classifications with more specific labels, therefore it is valuable to see what this adds to the predictions. To do this it is useful to compare it with a similar algorithm that does not use the WordTree in order to isolate the effect of the WordTree.

YOLO's WordTree uses the Region layer, a layer that, if not WordTree is used, assumes that classes are mutually exclusive [6]. This assumption does not hold for our dataset so comparing YOLO9000 with and without WordTree has some drawbacks. But since this is the most direct comparison possible, and since everything else is equal it is an effective way to isolate the improvements made by the WordTree. Both networks are trained in the same way as explained in the Training section.

Table 4: Performance of YOLO9000 with and without WordTree (WT).

Measure	YOLO9000	YOLO9000-WT
oLRP ↓	0.830	0.789
mAP ↑	0.610	**0.728**
Precision ↑	0.693	0.731
Recall ↑	0.488	0.556
s	0.460	0.520

Table 4 shows an oLRP improvement of 4.1% when using WordTree. Here ↓ represents a metric where a lower value is better. The localization of objects can be presumed to be equally well predicted, since the networks are the same. This shows that the classification is performing better as a result of the WordTree. This happens for a couple of reasons. Firstly the network is able to learn more from the data since the error is backpropagated for all classes above the predicted class in the WordTree. This results in a better-trained model. Secondly there is more context at detection time. The assigned class is the class furthest down in the WordTree that has a threshold above the hierarchical threshold value specified. This means that if a class is going to be applied, both the applied class and its ancestors needs to have high probabilities. Thus increasing the confidence and reducing faulty classifications.

YOLOv3 Architecture

YOLO's WordTree algorithm is originally only implemented in the YOLO9000 architecture. However YOLOv3 performs significantly better, with a larger and more complex network with 139.611 Giga Floating-Point Operations per Second (GFLOPS) [14] compared to 40.483 GFLOPS with our configurations. This experiment tests the possiblity of increasing the performance by creating an architecture that combines WordTree with YOLOv3. The network is changed to use softmax across sibling classes and use the loss function from YOLOv2. This means that while YOLOv3 is a multi-label detector, the modified YOLOv3 no longer is. This is because of changing the prediction layers from being YOLO layers to being region layers. The network change is illustrated with purple in figure 4.

Table 5 shows that YOLOv3 without WordTree performs better than YOLO9000 with WordTree with an improvement in oLRP of 7.6%. The addition of WordTree in YOLOv3 leads to the significant improvement of 14.9% to oLRP and 17.3% to mAP.

Introducing Soft-NMS

A drawback of the modification that is done to make NMS classless is that it removes YOLO's ability to keep bounding boxes for different fish that slightly overlap. By modifying the NMS algorithm to use Soft-NMS, as described in the Architecture Improvements

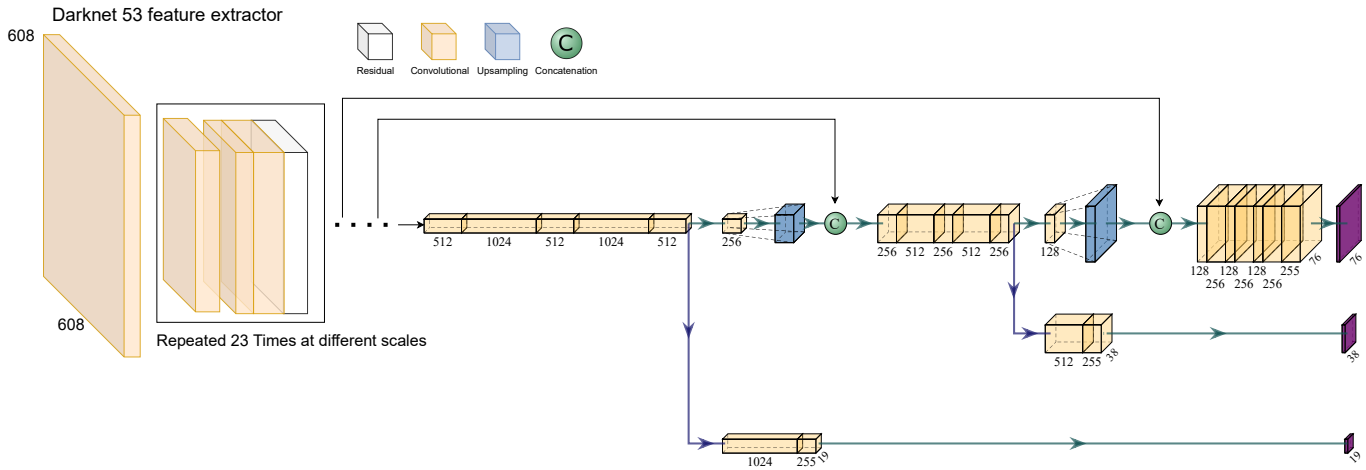


Figure 4: The network architecture. The purple layers are the YOLO detection layers which have been swapped out with region layers.

Table 5: Performance of YOLO9000-WT and YOLOv3 with and without WordTree.

Measure	9000-WT	v3	v3-WT
oLRP ↓	0.731	0.655	0.556
mAP ↑	0.726	0.793	**0.899**
Precision ↑	0.887	0.800	0.936
Recall ↑	0.627	0.770	0.807
s	0.550	0.380	0.710

section, suppressed bounding boxes have their probability decreased instead of being completely suppressed. This experiment tests the performance of YOLOv3-WT with soft-NMS.

Table 6: Performance improvement from Soft-NMS on YOLOv3.

Measure	YOLOv3-WT	YOLOv3-WT Soft-NMS
oLRP ↓	0.556	0.556
mAP ↑	0.899	**0.918**
Precision ↑	0.936	0.936
Recall ↑	0.807	0.807
s	0.710	0.710

Table 6 shows that Soft-NMS increases the mAP by 1.9%. This likely stems from overlapping boxes that come due to overlapping objects that are no longer removed. This is slightly higher than the improvements seen on the VOC 2007 and the COCO datasets [17]. However, no change is seen in oLRP, Precision or Recall. This is because Soft-NMS gives overlapping bounding boxes very low probabilities, far below the *s* threshold of 0.71. As a result they are removed before calculating oLRP, precision and recall as the probability threshold for boxes to be included, s, removes these boxes. Boxes with lower probabilities are removed, otherwise the network would produce far too many erroneous predictions and thus decrease performance.

While Soft-NMS improves the mAP score it would have no real impact on detection performance in normal use cases in this instance.

Discussion

The last experiment concludes the modifications to the network itself. After all the improvements made in the previous experiments the network has become significantly different than YOLO. The proposed object detector is now specifically tailored for fish detection and therefore named YOLO Fish. YOLO Fish is made publicly available on Github[2]. In this section, the results of the algorithm as a fish detector will be assessed. This section will present some new results and discuss them in the context of hierarchical classification.

The experiments shows the gradual improvements to YOLO, starting with YOLO9000 and resulting in YOLO Fish. The performance gains are summarized in Table 7. The improvements are a result of first utilizing WordTree, then modifying NMS to account for the bounding box problem. After this the network is modified to use YOLOv3 with WordTree. Lastly NMS is improved again with Soft-NMS. The result of this process is the proposed algorithm called YOLO Fish.

Table 8 shows performance of YOLO Fish at different IoU thresholds. It can be observed that the algorithm doesn't have the most accurate localization as the mAP quickly falls off as the IoU threshold for considering a detection correct is increased. Figure 7 shows that it is very good at classification. It can therefore be said to be better at classification than localization. This is one of the drawbacks of using detectors that use one

[2]The source code for YOLO Fish can be found on the Github repository https://github.com/orilan93/darknet

Table 7: Summary of incremental performance improvements.

	YOLO9000				YOLO Fish
Modified NMS	✓	✓		✓	✓
WordTree		✓		✓	✓
Yolov3 network			✓	✓	✓
Soft-NMS					✓
oLRP ↓	0.830	0.731	0.655	0.556	0.556
Precision ↑	0.693	0.877	0.800	0.936	0.936
Recall ↑	0.488	0.627	0.770	0.807	0.807
mAP ↑	0.610	0.728	0.793	0.899	**0.918**

networks for both localization and classification. This especially applies to YOLO as the network has large bounding box localization restrictions due to the bounding box predictions being based on a grid [6][18].

Table 8: Performance of YOLO Fish at different IoU thresholds.

Network	$mAP_{IoU=0.3}$	$mAP_{IoU=0.5}$	$mAP_{IoU=0.7}$
YOLO Fish	0.922	0.918	0.744

Figure 7 shows that the algorithm manages to correctly classify every fish except one of the fish that it detects. The mistake it makes is the misclassification of *Pollachius pollachius* as *Pollachius virens*, two very similar species. It is also often able to classify fish more specifically than the specified label for "Fish" and Gadidae, this means that the classifier possibly is better than humans at classification. However with the dataset that is used, there is no way to say whether the more specific labels are correct or not, so no definitive claim can be made.

The use of the WordTree adds an extra hyperparameter that needs to be set according to the use case of the model, a hierarchical probability threshold. This hierarchical probability threshold specifies how certain a classification must be for a specific class prediction before using the parent class. This is needed because the network will predict high probabilities for all classes in the branch of the hierarchy that the object is likely to belong to, and a cutoff value needs to be set so it can determine how deep in the tree the predictions will go. This hyperparameter is only used when predicting and not during training.

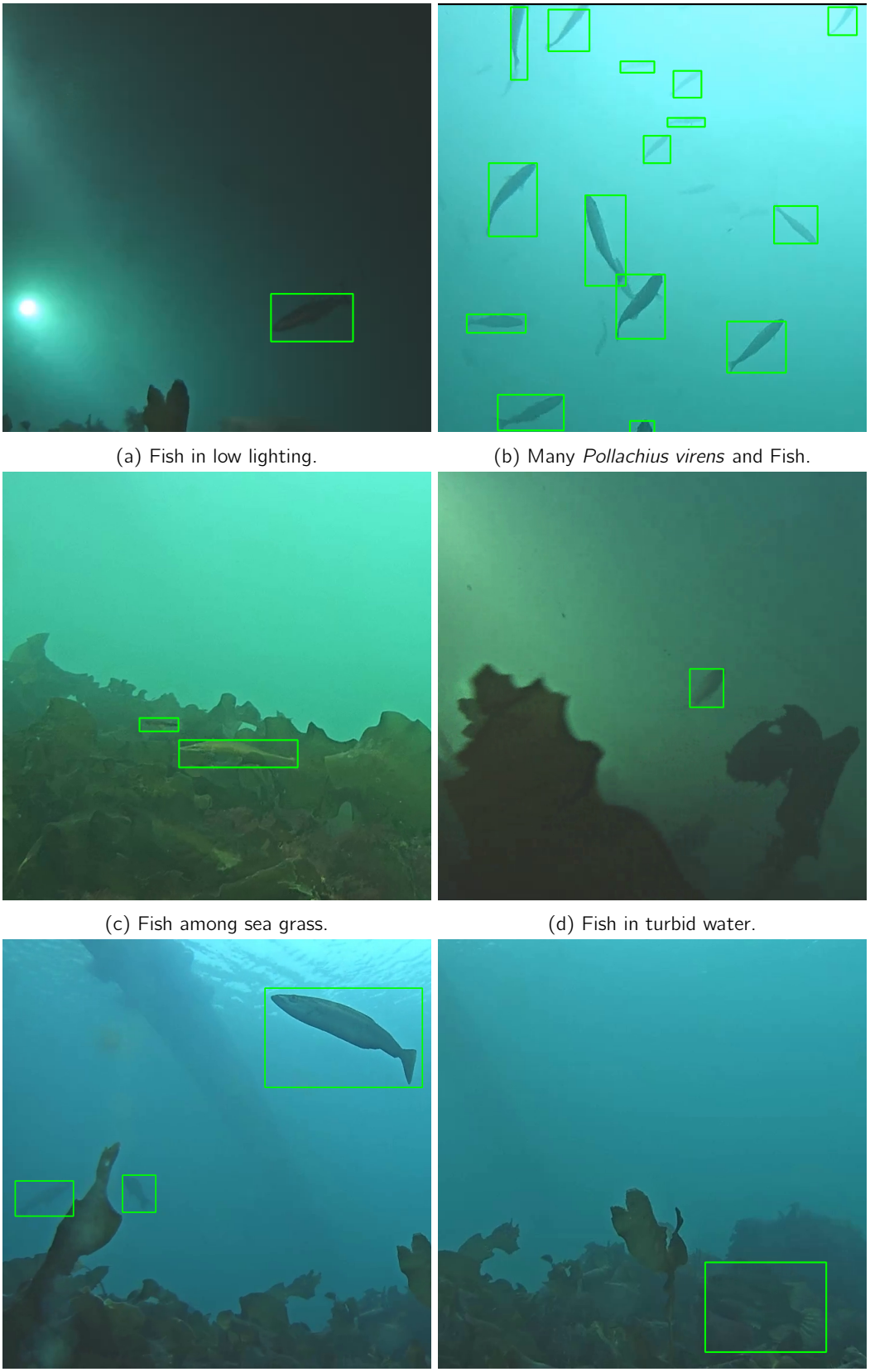
By using a hierarchical distance it becomes possible to quantitatively measure the effect of the hierarchical probability threshold. To do this hierarchical distance is defined to be the number of levels up or down in the hierarchy the ground truth is from the prediction. For instance if the algorithm predicts Gadidae and the ground truth is "Fish" the hierarchical distance is 1, or if the ground truth is Pollachius virens the hierarchical distance is -2.

Table 9 shows the average hierarchical distance for different hierarchical threshold levels for YOLO Fish. It can be noticed that the *average* hierarchical distance is quite high when the probability threshold is 0.2 or 0.5, and becomes lower at hierarchical thresholds of 0.99 and 0.999. At hierarchical threshold 1 all detections are classified as fish. This illustrates that increasing the probability threshold makes it possible to be very confident in the predicted species, at the cost of how specific the prediction is. For most hierarchical probability thresholds the network is able to classify species more confidently and specifically than a human, as evident by the positive average hierarchical distance for all species. However, this cannot be verified as correct.

Table 9: The *average* hierarchical distance from label to detection for different hierarchical threshold levels for YOLO Fish. The table only contains correct predictions.

	Hierarchical Probability Threshold				
Ground truth	1	0.999	0.99	0.5	0.2
Fish	0	2.20	3.36	3.89	3.92
Gadidae	-1	1.75	2.58	2.99	3.00
C. lumpus	-1	-0.14	0.00	0.00	0.00
L. bergylta	-2	-0.66	-0.67	0.00	0.00
C. rupestris	-2	-0.42	-0.13	0.00	0.00
P. virens	-3	-0.28	-0.06	0.00	0.00
P. pollachius	-3	-0.08	-0.08	0.00	0.00
All	-0.53	1.74	2.70	3.15	3.18

The ability for the end-user to set a hierarchical probability threshold makes the algorithm more versatile. For instance, if it is used to collect data for determining fishing quotas there might be a need to have very high confidence in the species that are identified. In this case a high hierarchical probability threshold might be used. Another use case might be to use it to show the public what species are in a particular area right now. The correctness in this instance might not be as important and a lower hierarchical probability threshold could be used.



(a) Fish in low lighting.

(b) Many *Pollachius virens* and Fish.

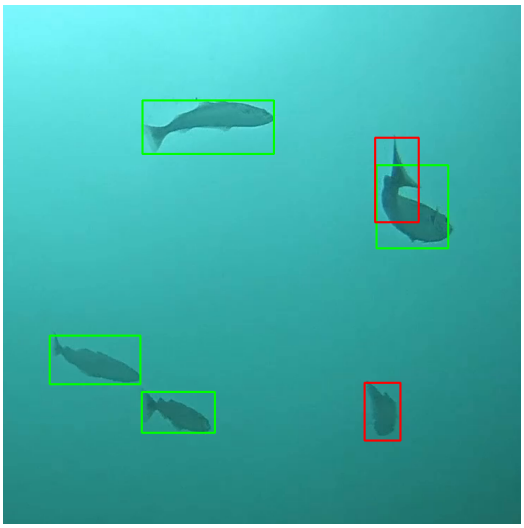
(c) Fish among sea grass.

(d) Fish in turbid water.

(e) "Fish" and Gadidae with low contrast.

(f) *Cyclopterus lumpus* among sea grass.

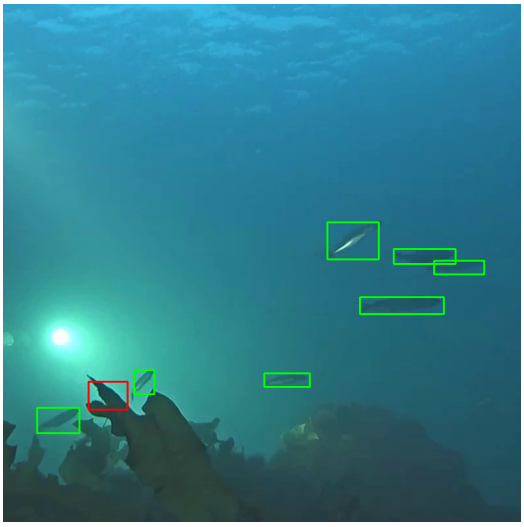
Figure 5: Good detections and classifications made by YOLO Fish. Green bounding box indicates correct class and location.



(a) Detector misclassifying species and producing a bad bounding box.

(b) Turbid water where the detector mistakes the debris for a fish.

(c) Detector missing obvious fish.



(d) Detector mistaking sea weed for fish.

Figure 6: Bad detections made by YOLO Fish.

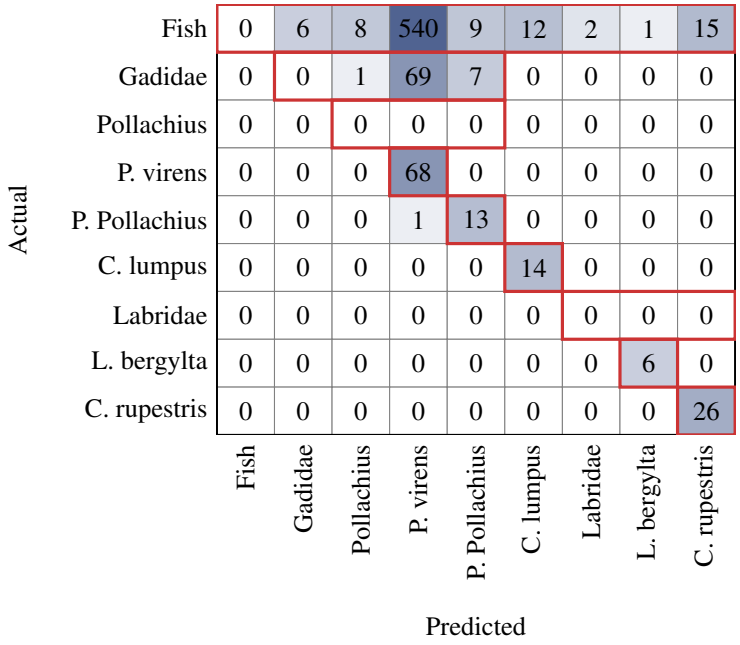


Figure 7: Hierarchical confusion matrix for YOLO Fish at an IoU of 0.5. All predictions that land inside the red boxes for each class is considered correct. If the prediction is inside the red box and to the right of the diagonal, the prediction is more specific than the ground truth label in the hierarchy.

So far the predictions have been discussed quantitatively, but discussing them in a qualitative manner compliments this and give insight. Examples of good and bad detections are in Figure 5 and 6 respectively. Figure 5 displays how the algorithm can perform well in varying lighting conditions and water clarity. It is however prone to make mistakes if there is a lot of debris in the water as in figure 6d which is captured in a storm. In situations where humans struggle with finding fish obscured by seaweed, the algorithm is able to detect them as seen in figure 5c and 5f. Sometimes it mistakes what clearly is seaweed for fish as in figure 6d. The shape of the fish is heavily relied upon for classification, and as can be seen in figure 6a where a Gadidae is misclassified as a *Cyclopterus lumpus* because of its round shape when swimming away from the camera in turbid water. However, when there is clear water the algorithm performs well, and can detect fish even with very low contrast as seen in figure 5e.

Conclusion

Identifying and classifying fish in video captured under water is a difficult task, as features are difficult to discern because of the water. This often leads to situations where the algorithms are unable to discriminate species. In these situations previous work applies a unknown class, and this reduces the available information for the network.

The objective of this work is to propose a technique to more confidently and accurately discriminate fish based on the available features. The proposed technique called YOLO Fish utilizes the inherent biological taxonomic tree of fish to perform hierarchical classification. This gives it the ability to apply higher hierarchical classes when it is uncertain, as it has learned what is common among all of the descendants of this higher class such as the large similarities among species and the nature of unclear underwater conditions. This novel technique makes both labeling the dataset easier, and increases the confidence in the classes applied.

YOLO Fish achieves the-state-of-the-art performance at object detection of Nordic fish species. YOLO Fish achieves an mAP of 91.8% on a dataset containing difficult conditions such as turbid water and nighttime pictures. This gives YOLO fish a significantly higher mAP than YOLOv3 which achieves an mAP of 79.3% with the same dataset and training. YOLO fish does this with an inference time of 26.4 ms per frame on a Tesla V100 GPU. This can be considered real-time with headroom.

Seeing that leveraging the similarities between species increases the accuracy and quality of predictions, it raises the question whether this technique can be applied in other cases. Further work is to investigate whether this technique can improve performance in other domains, such as insect or bacteria classification.

Conflict of interest

Authors state no conflict of interest.

Acknowledgements

Morten Goodwin and Aditya Gupta is supported by the Norwegian Research Council HAVBRUK2 innovation project CreateView Project nr. 309784.

References

1. Perry D, Staveley TA, and Gullström M. Habitat connectivity of fish in temperate shallow-water seascapes. Frontiers in Marine Science 2018; 4:440. Available from: https://www.frontiersin.org/articles/10.3389/fmars.2017.00440/full
2. Weinstein BG. A computer vision for animal ecology. Journal of Animal Ecology 2018; 87:533–45. Available from: https://doi.org/10.1111/1365-2656.12780
3. Stål J, Paulsen S, Pihl L, Rönnbäck P, Söderqvist T, and Wennhage H. Coastal habitat support to fish and fisheries in Sweden: Integrating ecosystem functions into fisheries management. Ocean & Coastal Management 2008; 51:594–600. Available from: https://doi.org/10.1016/j.ocecoaman.2008.06.006
4. Huang PX, Boom BJ, and Fisher RB. Underwater live fish recognition using a balance-guaranteed optimized tree. *Asian Conference on Computer Vision*. Springer. 2012 :422–33. Available from: https://link.springer.com/chapter/10.1007/978-3-642-37331-2_32
5. Phoenix X. Huang Bastiaan B. Boom RBF. "Fish Recognition Ground-Truth data. Accessed 20. Jan 2020. 2013 Sep. Available from: http://groups.inf.ed.ac.uk/f4k/GROUNDTRUTH/RECOG/
6. Redmon J and Farhadi A. YOLO9000: better, faster, stronger. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017 :7263–71. Available from: https://ieeexplore.ieee.org/document/8100173
7. Xiu Li, Min Shang, Qin H, and Liansheng Chen. Fast accurate fish detection and recognition of underwater images with Fast R-CNN. *OCEANS 2015 - MTS/IEEE Washington*. 2015 Oct :1–5. DOI: 10.23919/OCEANS.2015.7404464. Available from: https://ieeexplore.ieee.org/document/7404464
8. Li X, Tang Y, and Gao T. Deep but lightweight neural networks for fish detection. *OCEANS 2017-Aberdeen*. IEEE. 2017 :1–5. Available from: https://ieeexplore.ieee.org/document/8084961
9. Raza K and Hong S. Fast and Accurate Fish Detection Design with Improved YOLO-v3 Model and Transfer Learning. 2020. Available from: https://www.researchgate.net/publication/339640242_Fast_and_Accurate_Fish_Detection_Design_with_Improved_YOLO-v3_Model_and_Transfer_Learning
10. Reithaug A. Employing Deep Learning for Fish Recognition. 2018. Available from: https://bora.uib.no/bora-xmlui/handle/1956/19620
11. Olsvik E, Trinh CMD, Knausgård KM, Wiklund A, Sørdalen TK, Kleiven AR, Jiao L, and Goodwin M. Biometric Fish Classification of Temperate Species Using Convolutional Neural Network with Squeeze-and-Excitation. Ed. by Wotawa F, Friedrich G, Pill I, Koitz-Hristov R, and Ali M. Cham, 2019. Available from: https://arxiv.org/abs/1904.02768
12. Knausgård KM, Wiklund A, Sørdalen TK, Halvorsen K, Kleiven AR, Jiao L, and Goodwin M. Temperate Fish Detection and Classification: a Deep Learning based Approach. 2021. Available from: https://link.springer.com/article/10.1007/s10489-020-02154-9
13. Redmon J, Divvala SK, Girshick RB, and Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2015 :779–88. Available from: https://ieeexplore.ieee.org/document/7780460
14. Redmon J and Farhadi A. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 2018. Available from: https://arxiv.org/abs/1804.02767
15. Oksuz K, Cam BC, Akbas E, and Kalkan S. Localization Recall Precision (LRP): A New Performance Metric for Object Detection. 2018. arXiv: 1807.01696 [cs.CV]. Available from: https://arxiv.org/abs/1807.01696
16. Everingham M, Eslami SA, Van Gool L, Williams CK, Winn J, and Zisserman A. The pascal visual object classes challenge: A retrospective. International journal of computer vision 2015; 111:98–136. Available from: https://link.springer.com/article/10.1007/s11263-014-0733-5
17. Bodla N, Singh B, Chellappa R, and Davis LS. Soft-NMS–improving object detection with one line of code. *Proceedings of the IEEE international conference on computer vision*. 2017 :5561–9. Available from: https://ieeexplore.ieee.org/document/8237855
18. Zhao ZQ, Zheng P, Xu St, and Wu X. Object detection with deep learning: A review. IEEE transactions on neural networks and learning systems 2019; 30:3212–32. Available from: https://ieeexplore.ieee.org/document/8627998
19. Salman A, Jalal A, Shafait F, Mian A, Shortis M, Seager J, and Harvey E. Fish species classification in unconstrained underwater environments based on deep learning: Fish classification based on deep learning. Limnology and Oceanography: Methods 2016 May; 14. DOI: 10.1002/lom3.10113. Available from: https://aslopubs.onlinelibrary.wiley.com/doi/10.1002/lom3.10113
20. Siddiqui S, Salman A, Malik I, Shafait F, Mian A, Shortis M, and Harvey E. Automatic fish species classification in underwater videos: Exploiting pretrained deep neural network models to compensate for limited labelled data. ICES Journal of Marine Science 2017 May; 75. DOI: 10.1093/icesjms/fsx109. Available from: https://academic.oup.com/icesjms/article/75/1/374/3924506
21. Abbas Q, Ibrahim MEA, and Jaffar MA. A comprehensive review of recent advances on deep vision systems. Artificial Intelligence Review 2019 Jun; 52:39–76. DOI: 10.1007/s10462-018-9633-3. Available from: https://doi.org/10.1007/s10462-018-9633-3
22. Redmon J. Darknet: Open Source Neural Networks in C. http://pjreddie.com/darknet/. 2013–2016
23. Bochkovskiy A, Wang CY, and Liao HYM. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934 2020. Available from: https://arxiv.org/abs/2004.10934

24. Wang CY, Liao HYM, Yeh IH, Wu YH, Chen PY, and Hsieh JW. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. arXiv preprint arXiv:1911.11929 2019. Available from: https://ieeexplore.ieee.org/document/9150780
25. Goodfellow I, Bengio Y, and Courville A. Deep Learning. http://www.deeplearningbook.org. MIT Press, 2016. Chap. 9
26. Taxonomic rank. Accessed 5. march 2020. Available from: https://en.wikipedia.org/wiki/Taxonomic%5C_rank
27. Redmon J. Darknet Repository. Accessed 24. april 2020. Available from: github.com/pjreddie/darknet
28. Davis J and Goadrich M. The Relationship between Precision-Recall and ROC Curves. *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006 :233–40. DOI: 10.1145/1143844.1143874. Available from: https://doi.org/10.1145/1143844.1143874
29. Everingham M, Van Gool L, Williams CK, Winn J, and Zisserman A. The pascal visual object classes (voc) challenge. International journal of computer vision 2010; 88:303–38. Available from: https://link.springer.com/article/10.1007/s11263-009-0275-4
30. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, and Zitnick CL. Microsoft coco: Common objects in context. *European conference on computer vision*. Springer. 2014 :740–55. Available from: https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48
31. COCO Detection Evaluation. Accessed 12. April 2020. Available from: http://cocodataset.org/%5C#detection-eval
32. COCO 2017 Object Detection Task. Accessed 12. April 2020. Available from: http://cocodataset.org/%5C#detection-2017
33. Peffers K, Tuunanen T, Rothenberger MA, and Chatterjee S. A Design Science Research Methodology for Information Systems Research. Journal of Management Information Systems 2007; 24:45–77. DOI: 10.2753/MIS0742-1222240302. eprint: https://doi.org/10.2753/MIS0742-1222240302. Available from: https://doi.org/10.2753/MIS0742-1222240302
34. Hevner A and Chatterjee S. Design science research in information systems. *Design research in information systems*. Springer, 2010 :9–22. Available from: https://link.springer.com/book/10.1007/978-1-4419-5653-8
35. Huang PX, Boom BJ, and Fisher RB. GMM improves the reject option in hierarchical classification for fish recognition. *IEEE Winter Conference on Applications of Computer Vision*. IEEE. 2014 :371–6. Available from: https://ieeexplore.ieee.org/document/6836076
36. Deng J, Dong W, Socher R, Li LJ, Li K, and Fei-Fei L. Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009 :248–55. Available from: https://ieeexplore.ieee.org/document/5206848
37. Marszalek M and Schmid C. Semantic hierarchies for visual object recognition. *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007 :1–7. Available from: https://ieeexplore.ieee.org/document/4270297
38. Marszałek M and Schmid C. Constructing category hierarchies for visual recognition. *European conference on computer vision*. Springer. 2008 :479–91. Available from: https://link.springer.com/chapter/10.1007/978-3-540-88693-8_35
39. Zweig A and Weinshall D. Exploiting object hierarchy: Combining models from different category levels. *2007 IEEE 11th International Conference on Computer Vision*. IEEE. 2007 :1–8. Available from: https://ieeexplore.ieee.org/document/4409064
40. Nister D and Stewenius H. Scalable recognition with a vocabulary tree. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. Ieee. 2006 :2161–8. Available from: https://ieeexplore.ieee.org/document/1641018
41. Silla CN and Freitas AA. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery 2011; 22:31–72. Available from: https://link.springer.com/article/10.1007/s10618-010-0175-9
42. Girshick R, Donahue J, Darrell T, and Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014 :580–7. Available from: https://ieeexplore.ieee.org/document/6909475
43. He K, Zhang X, Ren S, and Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence 2015; 37:1904–16. Available from: https://ieeexplore.ieee.org/document/7005506
44. Girshick R. Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*. 2015 :1440–8. Available from: https://ieeexplore.ieee.org/document/7410526
45. Ren S, He K, Girshick R, and Sun J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*. 2015 :91–9. Available from: https://ieeexplore.ieee.org/abstract/document/7485869
46. Lin TY, Dollár P, Girshick R, He K, Hariharan B, and Belongie S. Feature pyramid networks for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017 :2117–25. Available from: https://ieeexplore.ieee.org/document/8099589
47. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, and Berg AC. Ssd: Single shot multibox detector. *European conference on computer vision*. Springer. 2016 :21–37. Available from: https://link.springer.com/chapter/10.1007/978-3-319-46448-0_2

48. Fu CY, Liu W, Ranga A, Tyagi A, and Berg AC. Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659 2017. Available from: https://arxiv.org/abs/1701.06659

49. Shen Z, Liu Z, Li J, Jiang YG, Chen Y, and Xue X. Dsod: Learning deeply supervised object detectors from scratch. *Proceedings of the IEEE international conference on computer vision*. 2017 :1919–27. Available from: https://ieeexplore.ieee.org/document/8237474

50. Lin TY, Goyal P, Girshick R, He K, and Dollár P. Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*. 2017 :2980–8. Available from: https://ieeexplore.ieee.org/document/8237586

51. Dai J, Li Y, He K, and Sun J. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems*. 2016 :379–87. Available from: https://proceedings.neurips.cc/paper/2016/hash/325995af77a0e8b06d1204a171010b3a-Abstract.html

52. Ghiasi G, Lin TY, and Le QV. Nas-fpn: Learning scalable feature pyramid architecture for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019 :7036–45. Available from: https://ieeexplore.ieee.org/document/8954436

53. He K, Zhang X, Ren S, and Sun J. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016 :770–8. Available from: https://ieeexplore.ieee.org/document/7780459

54. Bewley A, Ge Z, Ott L, Ramos F, and Upcroft B. Simple online and realtime tracking. *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. 2016 :3464–8. Available from: https://ieeexplore.ieee.org/document/7533003

55. Wojke N, Bewley A, and Paulus D. Simple online and realtime tracking with a deep association metric. *2017 IEEE international conference on image processing (ICIP)*. IEEE. 2017 :3645–9. Available from: https://ieeexplore.ieee.org/document/8296962

56. Bathija A and Sharma G. Visual Object Detection and Tracking using YOLO and SORT. 2019

57. Yu F, Li W, Li Q, Liu Y, Shi X, and Yan J. Poi: Multiple object tracking with high performance detection and appearance feature. *European Conference on Computer Vision*. Springer. 2016 :36–42. Available from: https://link.springer.com/chapter/10.1007/978-3-319-48881-3_3

58. Zhang H and Wang N. On the stability of video detection and tracking. arXiv preprint arXiv:1611.06467 2016. Available from: https://arxiv.org/abs/1611.06467

59. Smeulders AW, Chu DM, Cucchiara R, Calderara S, Dehghan A, and Shah M. Visual tracking: An experimental survey. IEEE transactions on pattern analysis and machine intelligence 2013; 36:1442–68. Available from: https://ieeexplore.ieee.org/document/6671560

60. Bernardin K and Stiefelhagen R. Evaluating multiple object tracking performance: the CLEAR MOT metrics. EURASIP Journal on Image and Video Processing 2008; 2008:1–10. Available from: https://link.springer.com/content/pdf/10.1155%2F2008%2F246309.pdf

61. Liu S, Qi L, Qin H, Shi J, and Jia J. Path aggregation network for instance segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018 :8759–68. Available from: https://ieeexplore.ieee.org/document/8579011

62. Kavukcuoglu K, Sermanet P, Boureau YL, Gregor K, Mathieu M, and Cun YL. Learning convolutional feature hierarchies for visual recognition. *Advances in neural information processing systems*. 2010 :1090–8. Available from: https://proceedings.neurips.cc/paper/2010/hash/a01610228fe998f515a72dd730294d87-Abstract.html

63. Maas AL, Hannun AY, and Ng AY. Rectifier nonlinearities improve neural network acoustic models. *Proc. icml*. Vol. 30. 2013 :3

64. He K, Zhang X, Ren S, and Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceedings of the IEEE international conference on computer vision*. 2015 :1026–34. Available from: https://ieeexplore.ieee.org/document/7410480

65. Fisher R, Chen-Burger J, Giordano D, Hardman L, and Lin FP. Fish4Knowledge: Collecting and Analyzing Massive Coral Reef Fish Video Data. Vol. 104. 2016 Jan. DOI: 10.1007/978-3-319-30208-9